EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	1247	711/165	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 08:22
L2	3510	711/170	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 08:22
L3	1141	711/171	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 08:22
L4	715	711/172	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 08:22
L5	1781	711/173	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 08:22
L6	393	717/141	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 08:22
L7	259	717/153	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 08:22
L8	443	717/148	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 08:22
L9	812	717/127	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 08:22
L10	3	(L1 or L2 or L3 or L4 or L5 or L6 or L7 or L8 or L9) and (heap with (suspend$4 near3 (thread or task)))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 08:24

EAST Search History

L11	0	(L1 or L2 or L3 or L4 or L5 or L6 or L7 or L8 or L9) and (heap with (compensation adj publish$6))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 10:23
L12	2257	707/205	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 10:23
L13	1324	707/206	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 10:23
L14	2	(L12 or L13) and ((stack adj allocation) with heap)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 10:35
S1	36	(william.xa. and wood.xa. and ("717"/$.ccls. or "712"/$.ccls.))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/10 08:18
S3	3	(escape adj analysis) and ((dynamic$4 near3 allocat$6) with stack)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/29 17:15
S4	11	("20020178437"\|"6170083"\|"6237141"\|"6332216"\|"6505344").PN.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/29 17:15
S5	0	("7076773").PN.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/06 16:54

EAST Search History

S6	101	("6457023" "6681385" "6381738" "7058943" "5948113" "6173421" "6505344" "6675378" "6253215" "6799191" "5408650" "6070173" "7032221" "6625808" "6415302" "6424977" "6434576" "6434577" "6449626" "5632034" "5819304" "5848423" "5893121" "5930807" "6038572" "6125434" "6289360" "6839725" "6862674" "4769771" "5560003" "5692185" "5579520" "6094664" "4907151" "5900001" "5903900" "5911144" "5915255" "5920876" "6049810" "6115782" "6327606" "6158024" "6999980" "5809554" "5822590" "5838977" "6381735" "6658652").pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/03 11:30
S7	37	S6 and (stack with heap)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/03 11:31
S8	18	("5481708"\|"5530870"\|"5590332"\|"6305009"\|"6381738"\|"6505344"\|"6675378"\|"6701520").PN.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 16:22
S9	406	allocation near3 heap	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 16:54
S10	30	(allocation near3 heap) with (recover$6 or return$6 or reinstat$6)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 17:16
S11	2	(allocation near3 heap) with ((thread or task) near3 (lock$4 or synchroniz$6))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 17:19
S12	1247	711/165	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 17:20

EAST Search History

S13	3510	711/170	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 17:20
S14	1141	711/171	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 17:20
S15	715	711/172	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 17:20
S16	1781	711/173	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 17:20
S17	393	717/141	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 17:21
S18	259	717/153	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 17:21
S19	443	717/148	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 17:21
S20	812	717/127	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/07/06 17:21

IEEE Xplore RELEASE 2.1

Results for "(('recover allocation' heap)<in>metadata)"

e-mail

Your search matched **0** documents.

A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History

New Search

Modify Search

(('recover allocation' heap)<in>metadata) Search

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages if you need assistance revising your search

© Copyright 2006 IE

Indexed by Inspec®

IEEE Xplore® RELEASE 2.1

Results for "(compensation publishing<in>metadata)"

e-mail

Your search matched 0 documents.

A maximum of 100 results are displayed, 25 to a page, sorted by Relevance in Descending order.

» Search Options

View Session History

New Search

Modify Search

(compensation publishing<in>metadata) Search

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages if you need assistance revising your search

© Copyright 2006 IE

Indexed by Inspec®

IEEE Xplore RELEASE 2.1

e-mail

Results for "('escape analysis' heap return<in>metadata)"

Your search matched 0 documents.

A maximum of 100 results are displayed, 25 to a page, sorted by Relevance in Descending order.

» Search Options

View Session History

New Search

Modify Search

('escape analysis' heap return<in>metadata) Search

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages if you need assistance revising your search

© Copyright 2006 IE

Indexed by Inspec

IEEE Xplore® RELEASE 2.1

Home | Login | Logout | Access Information | Ale

Welcome United States Patent and Trademark Office

Search Results BROWSE SEARCH IEEE XPLORE GUIDE

Results for "('escape analysis'<in>metadata)" e-mail

Your search matched **6** of **1372086** documents.

A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History

New Search

Modify Search

('escape analysis'<in>metadata) Search

Check to search only within this results set

» Key

Display Format: Citation Citation & Abstract

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

view selected items Select All Deselect All

1. **Putting escape analysis to work for software testing**
Souter, A.L.; Pollock, L.L.;
Software Maintenance, 2002. Proceedings. International Conference on
3-6 Oct. 2002 Page(s):430 - 439
Digital Object Identifier 10.1109/ICSM.2002.1167800
AbstractPlus | Full Text: PDF(333 KB) IEEE CNF
Rights and Permissions

2. **Runtime analysis of atomicity for multithreaded programs**
Wang, L.; Stoller, S.D.;
Software Engineering, IEEE Transactions on
Volume 32, Issue 2, Feb. 2006 Page(s):93 - 110
Digital Object Identifier 10.1109/TSE.2006.1599419
AbstractPlus | Full Text: PDF(3000 KB) IEEE JNL
Rights and Permissions

3. **Tracking object life cycle for leakage energy optimization**
Chen, G.; Vijaykrishnan, N.; Kandemir, M.; Irwin, M.J.; Wolczko, M.;
Hardware/Software Codesign and System Synthesis, 2003. First IEEE/ACM/IFIP International Con
1-3 Oct. 2003 Page(s):213 - 218
AbstractPlus | Full Text: PDF(539 KB) IEEE CNF
Rights and Permissions

4. **Compiler optimized remote method invocation**
Veldema, R.; Philippsen, M.;
Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on
2003 Page(s):127 - 136
Digital Object Identifier 10.1109/CLUSTR.2003.1253308
AbstractPlus | Full Text: PDF(365 KB) IEEE CNF
Rights and Permissions

5. **The Pensieve project: a compiler infrastructure for memory models**
Chi-Leung Wong; Zehra Sura; Xing Fang; Midkiff, S.P.; Jaejin Lee; Padua, D.;
Parallel Architectures, Algorithms and Networks, 2002. I-SPAN '02. Proceedings. International Syn
22-24 May 2002 Page(s):209 - 214
Digital Object Identifier 10.1109/ISPAN.2002.1004288
AbstractPlus | Full Text: PDF(230 KB) IEEE CNF

Rights and Permissions

6. **A generic worklist algorithm for graph reachability problems in program analysis**
Rayside, D.; Kontogiannis, K.;
Software Maintenance and Reengineering, 2002. Proceedings. Sixth European Conference on
11-13 March 2002 Page(s):67 - 76
Digital Object Identifier 10.1109/CSMR.2002.995791
AbstractPlus | Full Text: PDF(381 KB) IEEE CNF
Rights and Permissions

Indexed by Inspec®

Help Contact Us Privac

© Copyright 2006 IE

PORTAL
USPTO
Subscribe (Full Service) Register (Limited Service, Free) Login
Search: The ACM Digital Library The Guide
"escape analysis"
SEARCH
THE ACM DIGITAL LIBRARY
Feedback Report a problem Satisfaction survey

Terms used **escape analysis** Found **204** of **178,880**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Escape analysis for Java™: Theory and practice

Bruno Blanchet
November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6
Publisher: ACM Press

Full text available: pdf(684.21 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Escape analysis is a static analysis that determines whether the lifetime of data may exceed its static scope.This paper first presents the design and correctness proof of an escape analysis for Java™. This analysis is interprocedural, context sensitive, and as flow-sensitive as the static single assignment form. So, assignments to object fields are analyzed in a flow-insensitive manner. Since Java is an imperative language, the effect of assignments must be precisely determined. Thi ...

Keywords: Java, optimization, stack allocation, static analysis, synchronization elimination

2 Stack allocation and synchronization optimizations for Java using escape analysis

Jong-Deok Choi, Manish Gupta, Mauricio J. Serrano, Vugranam C. Sreedhar, Samuel P. Midkiff
November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6
Publisher: ACM Press

Full text available: pdf(632.85 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This article presents an *escape analysis* framework for Java to determine (1) if an object is not reachable after its method of creation returns, allowing the object to be allocated on the stack, and (2) if an object is reachable only from a single thread during its lifetime, allowing unnecessary synchronization operations on that object to be removed. We introduce a new program abstraction for escape analysis, the *connection graph*, that is used to establish reachability relationshi ...

Keywords: Connection graphs, escape analysis, points-to graph

3 Escape analysis on lists

Young Gil Park, Benjamin Goldberg
July 1992 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1992 conference on Programming language design and implementation PLDI '92**, Volume 27 Issue 7
Publisher: ACM Press

Full text available: pdf(1.08 MB) Additional Information: full citation, abstract, references, citings, index terms

Higher order functional programs constantly allocate objects dynamically. These objects are typically cons cells, closures, and records and are generally allocated in the heap and reclaimed later by some garbage collection process. This paper describes a compile time analysis, called escape analysis, for determining the lifetime of dynamically created objects in higher order functional programs, and describes optimizations that can be performed, based on the analysis, to improve storage all ...

4 Escape analysis for Java

Jong-Deok Choi, Manish Gupta, Mauricio Serrano, Vugranam C. Sreedhar, Sam Midkiff
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10
Publisher: ACM Press

Full text available: pdf(1.85 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a simple and efficient data flow algorithm for escape analysis of objects in Java programs to determine (i) if an object can be allocated on the stack; (ii) if an object is accessed only by a single thread during its lifetime, so that synchronization operations on that object can be removed. We introduce a new program abstraction for escape analysis, the connection graph, that is used to establish reachability relationships between objects and object ref ...

5 Escape analysis for object-oriented languages: application to Java

Bruno Blanchet
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10
Publisher: ACM Press

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

Escape analysis [27, 14, 5] is a static analysis that determines whether the lifetime of data exceeds its static scope.The main originality of our escape analysis is that it determines precisely the effect of assignments, which is necessary to apply it to object oriented languages with promising results, whereas previous work [27, 14, 5] applied it to functional languages and were very imprecise on assignments. Our implementation analyses the full Java™ Language. ...

6 Dynamic compilation techniques: Escape analysis in the context of dynamic compilation and deoptimization

Thomas Kotzmann, Hanspeter Mössenböck
June 2005 **Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments**
Publisher: ACM Press

Full text available: pdf(203.40 KB) Additional Information: full citation, abstract, references, index terms

In object-oriented programming languages, an object is said to *escape* the method or thread in which it was created if it can also be accessed by other methods or threads. Knowing which objects do not escape allows a compiler to perform aggressive

optimizations.This paper presents a new intraprocedural and interprocedural algorithm for escape analysis in the context of dynamic compilation where the compiler has to cope with dynamic class loading and deoptimization. It was implemented for S ...

Keywords: Java, deoptimization, escape analysis, just-in-time compilation, optimization, scalar replacement, stack allocation, synchronization removal

7 Compositional pointer and escape analysis for Java programs

John Whaley, Martin Rinard

October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10

Publisher: ACM Press

Full text available: pdf(2.42 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a combined pointer and escape analysis algorithm for Java programs. The algorithm is based on the abstraction of points-to escape graphs, which characterize how local variables and fields in objects refer to other objects. Each points-to escape graph also contains escape information, which characterizes how objects allocated in one region of the program can escape to be accessed by another region. The algorithm is designed to analyze arbitrary regions of complete or inco ...

8 Reference escape analysis: optimizing reference counting based on the lifetime of references

Young Gil Park, Benjamin Goldberg

May 1991 **ACM SIGPLAN Notices , Proceedings of the 1991 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation PEPM '91**, Volume 26 Issue 9

Publisher: ACM Press

Full text available: pdf(1.06 MB) Additional Information: full citation, references, citings, index terms

9 Pointer and escape analysis for multithreaded programs

Alexandru Salcianu, Martin Rinard

June 2001 **ACM SIGPLAN Notices , Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming PPoPP '01**, Volume 36 Issue 7

Publisher: ACM Press

Full text available: pdf(318.11 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a new combined pointer and escape analysis for multithreaded programs. The algorithm uses a new abstraction called parallel interaction graphs to analyze the interactions between threads and extract precise points-to, escape, and action ordering information for objects accessed by multiple threads. The analysis is compositional, analyzing each method or thread once to extract a parameterized analysis result that can be specialized for use in any context. ...

10 Escape analysis: correctness proof, implementation and experimental results

Bruno Blanchet

January 1998 **Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Publisher: ACM Press

Full text available: pdf(1.68 MB) Additional Information: full citation, references, citings, index terms

11 Incrementalized pointer and escape analysis

Frédéric Vivien, Martin Rinard
May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation PLDI '01**, Volume 36 Issue 5
Publisher: ACM Press

Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a new pointer and escape analysis. Instead of analyzing the whole program, the algorithm incrementally analyzes only those parts of the program that may deliver useful results. An analysis policy monitors the analysis results to direct the incremental investment of analysis resources to those parts of the program that offer the highest expected optimization return.

Our experimental results show that almost all of the objects are allocated at a small number of allocation sit ...

12 Access control analysis: Data-centric security: role analysis and role typestates

Vugranam C. Sreedhar
June 2006 **Proceedings of the eleventh ACM symposium on Access control models and technologies SACMAT '06**
Publisher: ACM Press
Full text available: pdf(270.98 KB) Additional Information: full citation, abstract, references, index terms

In J2EE and .NET roles are assigned to methods using external configuration files, called the deployment descriptors. Assigning roles to methods, although conceptually simple, in practice it is quite complicated. For instance, in order for a deployer to assign a role *r* to a method *m*, the deployer must understand the set of roles *R* that are assigned to each method *n* that can be invoked directly or indirectly from *m*, and that *r* has to be "consistently" ass ...

Keywords: RBAC, role analysis, role escape analysis, role typestates

13 OMEN: A strategy for testing object-oriented software

Amie L. Souter, Lori L. Pollock
August 2000 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2000 ACM SIGSOFT international symposium on Software testing and analysis ISSTA '00**, Volume 25 Issue 5
Publisher: ACM Press

Full text available: pdf(329.36 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a strategy for structural testing of object-oriented software systems with possibly unknown clients and unknown information about invoked methods. By exploiting the combined points-to and escape analysis developed for compiler optimization, our testing paradigm does not require a whole program representation to be in memory simultaneously for testing analysis. Potential effects from outside the component under test are easily identified and reported to the tester. As c ...

14 Parallel architectures: Automatic fence insertion for shared memory multiprocessing

Xing Fang, Jaejin Lee, Samuel P. Midkiff
June 2003 **Proceedings of the 17th annual international conference on Supercomputing**

Publisher: ACM Press

Full text available: pdf(327.89 KB) Additional Information: full citation, abstract, references, index terms

In general, the hardware memory consistency model in a multiprocessor system is not identical to the memory model at the programming language level. Consequently, the programming language memory model must be mapped onto the hardware memory model. Memory fence instructions can be inserted by the compiler where needed to accomplish this mapping. We have developed and implemented several fence insertion and optimization algorithms in our Pensieve compiler project. We present the different fence in ...

Keywords: Java, compilers, consistency models

15 Field analysis: getting useful and low-cost interprocedural information

Sanjay Ghemawat, Keith H. Randall, Daniel J. Scales

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5

Publisher: ACM Press

Full text available: pdf(686.96 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a new limited form of interprocedural analysis called field analysis that can be used by a compiler to reduce the costs of modern language features such as object-oriented programming, automatic memory management, and run-time checks required for type safety. Unlike many previous interprocedural analyses, our analysis is cheap, and does not require access to the entire program. Field analysis exploits the declared access restrictions placed on fields in a modul ...

16 Partial method compilation using dynamic profile information

John Whaley

October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '01**, Volume 36 Issue 11

Publisher: ACM Press

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

The traditional tradeoff when performing dynamic compilation is that of fast compilation time versus fast code performance. Most dynamic compilation systems for Java perform selective compilation and/or optimization at a method granularity. This is the not the optimal granularity level. However, compiling at a sub-method granularity is thought to be too complicated to be practical. This paper describes a straightforward technique for performing compilation and optimizations at a finer, sub-metho ...

17 Compiler optimizations for power, performance: Tracking object life cycle for leakage energy optimization

G. Chen, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, M. Wolczko

October 2003 **Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis**

Publisher: ACM Press

Full text available: pdf(304.66 KB) Additional Information: full citation, abstract, references, index terms

The focus of this work is on utilizing the state of objects during their lifespan in optimizing the leakage energy consumed in the data caches when executing embedded Java applications. Our analysis reveals that a major portion of the leakage energy is actually wasted in retaining the objects beyond their last use. In order to eliminate this wastage,

we investigate three approaches that use the garbage collector, escape analysis and last use analysis for reducing leakage energy. Finally, we trac ...

Keywords: Java, cache, leakage energy

18 A region-based compilation technique for a Java just-in-time compiler

Toshio Suganuma, Toshiaki Yasue, Toshio Nakatani

May 2003 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation PLDI '03**, Volume 38 Issue 5

Publisher: ACM Press

Full text available: pdf(158.62 KB) Additional Information: full citation, abstract, references, citings, index terms

Method inlining and data flow analysis are two major optimization components for effective program transformations, however they often suffer from the existence of rarely or never executed code contained in the target method. One major problem lies in the assumption that the compilation unit is partitioned at method boundaries. This paper describes the design and implementation of a region-based compilation technique in our dynamic compilation system, in which the compiled regions are selected a ...

Keywords: dynamic compilers, on-stack replacement, partial inlining, region-based compilation

19 Cloning-based context-sensitive pointer alias analysis using binary decision diagrams

John Whaley, Monica S. Lam

June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation PLDI '04**, Volume 39 Issue 6

Publisher: ACM Press

Full text available: pdf(277.87 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents the first scalable context-sensitive, inclusion-based pointer alias analysis for Java programs. Our approach to context sensitivity is to create a clone of a method for every context of interest, and run a *context-insensitive* algorithm over the expanded call graph to get *context-sensitive* results. For precision, we generate a clone for every acyclic path through a program's call graph, treating methods in a strongly connected component as a single node. Normally ...

Keywords: Datalog, Java, binary decision diagrams, cloning, context-sensitive, inclusion-based, logic programming, pointer analysis, program analysis, scalable

20 Effective synchronization removal for Java

Erik Ruf

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5

Publisher: ACM Press

Full text available: pdf(819.53 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a new technique for removing unnecessary synchronization operations from statically compiled Java programs. Our approach improves upon current efforts based on escape analysis, as it can eliminate synchronization operations even on objects that

escape their allocating threads. It makes use of a compact, equivalence-class-based representation that eliminates the need for fixed point operations during the analysis. We describe and evaluate the performance of an implemen ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2006 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

PORTAL
USPTO
Subscribe (Full Service) Register (Limited Service, Free) Login
Search: The ACM Digital Library The Guide
"escape analysis" "open-world"
SEARCH
THE ACM DIGITAL LIBRARY
Feedback Report a problem Satisfaction survey

Terms used **escape analysis open world** Found **238** of **178,880**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Automatic detection of immutable fields in Java
Sara Porat, Marina Biberstein, Larry Koved, Bilha Mendelson
November 2000 **Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research**
Publisher: IBM Press
Full text available: pdf(300.59 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper introduces techniques to detect mutability of fields and classes in Java. A variable is considered to be mutable if a new value is stored into it, as well as if any of its reachable variables is mutable. We present a static flow-sensitive analysis algorithm which can be applied to any Java component. The analysis classifies fields and classes as either mutable or immutable. In order to facilitate openworld analysis, the algorithm identifies situations that expose variables to potentia ...

2 Ownership, encapsulation and the disjointness of type and effect
Dave Clarke, Sophia Drossopoulou
November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '02**, Volume 37 Issue 11
Publisher: ACM Press
Full text available: pdf(475.72 KB) Additional Information: full citation, abstract, references, citings, index terms

Ownership types provide a statically enforceable notion of object-level encapsulation. We extend ownership types with computational effects to support reasoning about object-oriented programs. The ensuing system provides both access control and effects reporting. Based on this type system, we codify two formal systems for reasoning about aliasing and the disjointness of computational effects. The first can be used to prove that evaluation of two expressions will never lead to aliases, while the ...

Keywords: aliasing, encapsulation, ownership types, type-and-effects systems

3 Escape analysis for Java™: Theory and practice
Bruno Blanchet
November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6
Publisher: ACM Press

Full text available: pdf(684.21 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Escape analysis is a static analysis that determines whether the lifetime of data may exceed its static scope.This paper first presents the design and correctness proof of an escape analysis for Java™. This analysis is interprocedural, context sensitive, and as flow-sensitive as the static single assignment form. So, assignments to object fields are analyzed in a flow-insensitive manner. Since Java is an imperative language, the effect of assignments must be precisely determined. Thi ...

Keywords: Java, optimization, stack allocation, static analysis, synchronization elimination

4 Stack allocation and synchronization optimizations for Java using escape analysis

Jong-Deok Choi, Manish Gupta, Mauricio J. Serrano, Vugranam C. Sreedhar, Samuel P. Midkiff

November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6

Publisher: ACM Press

Full text available: pdf(632.85 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This article presents an *escape analysis* framework for Java to determine (1) if an object is not reachable after its method of creation returns, allowing the object to be allocated on the stack, and (2) if an object is reachable only from a single thread during its lifetime, allowing unnecessary synchronization operations on that object to be removed. We introduce a new program abstraction for escape analysis, the *connection graph*, that is used to establish reachability relationshi ...

Keywords: Connection graphs, escape analysis, points-to graph

5 Escape analysis on lists

Young Gil Park, Benjamin Goldberg

July 1992 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1992 conference on Programming language design and implementation PLDI '92**, Volume 27 Issue 7

Publisher: ACM Press

Full text available: pdf(1.08 MB) Additional Information: full citation, abstract, references, citings, index terms

Higher order functional programs constantly allocate objects dynamically. These objects are typically cons cells, closures, and records and are generally allocated in the heap and reclaimed later by some garbage collection process. This paper describes a compile time analysis, called escape analysis, for determining the lifetime of dynamically created objects in higher order functional programs, and describes optimizations that can be performed, based on the analysis, to improve storage all ...

6 Escape analysis for Java

Jong-Deok Choi, Manish Gupta, Mauricio Serrano, Vugranam C. Sreedhar, Sam Midkiff

October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10

Publisher: ACM Press

Full text available: pdf(1.85 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a simple and efficient data flow algorithm for escape analysis of objects in Java programs to determine (i) if an object can be allocated on the stack; (ii) if an object is accessed only by a single thread during its lifetime, so that synchronization operations on that object can be removed. We introduce a new program abstraction for escape analysis, the connection graph, that is used to establish reachability relationships between objects and object ref ...

7 Escape analysis for object-oriented languages: application to Java

Bruno Blanchet

October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10

Publisher: ACM Press

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

Escape analysis [27, 14, 5] is a static analysis that determines whether the lifetime of data exceeds its static scope.The main originality of our escape analysis is that it determines precisely the effect of assignments, which is necessary to apply it to object oriented languages with promising results, whereas previous work [27, 14, 5] applied it to functional languages and were very imprecise on assignments. Our implementation analyses the full Java™ Language. ...

8 Dynamic compilation techniques: Escape analysis in the context of dynamic compilation and deoptimization

Thomas Kotzmann, Hanspeter Mössenböck

June 2005 **Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments**

Publisher: ACM Press

Full text available: pdf(203.40 KB) Additional Information: full citation, abstract, references, index terms

In object-oriented programming languages, an object is said to *escape* the method or thread in which it was created if it can also be accessed by other methods or threads. Knowing which objects do not escape allows a compiler to perform aggressive optimizations.This paper presents a new intraprocedural and interprocedural algorithm for escape analysis in the context of dynamic compilation where the compiler has to cope with dynamic class loading and deoptimization. It was implemented for S ...

Keywords: Java, deoptimization, escape analysis, just-in-time compilation, optimization, scalar replacement, stack allocation, synchronization removal

9 Compositional pointer and escape analysis for Java programs

John Whaley, Martin Rinard

October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10

Publisher: ACM Press

Full text available: pdf(2.42 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a combined pointer and escape analysis algorithm for Java programs. The algorithm is based on the abstraction of points-to escape graphs, which characterize how local variables and fields in objects refer to other objects. Each points-to escape graph also contains escape information, which characterizes how objects allocated in one region of the program can escape to be accessed by another region. The algorithm is designed to analyze arbitrary regions of complete or inco ...

10 Reference escape analysis: optimizing reference counting based on the lifetime of references
Young Gil Park, Benjamin Goldberg
May 1991 **ACM SIGPLAN Notices , Proceedings of the 1991 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation PEPM '91**, Volume 26 Issue 9
Publisher: ACM Press
Full text available: pdf(1.06 MB) Additional Information: full citation, references, citings, index terms

11 Pointer and escape analysis for multithreaded programs
Alexandru Salcianu, Martin Rinard
June 2001 **ACM SIGPLAN Notices , Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming PPoPP '01**, Volume 36 Issue 7
Publisher: ACM Press
Full text available: pdf(318.11 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a new combined pointer and escape analysis for multithreaded programs. The algorithm uses a new abstraction called parallel interaction graphs to analyze the interactions between threads and extract precise points-to, escape, and action ordering information for objects accessed by multiple threads. The analysis is compositional, analyzing each method or thread once to extract a parameterized analysis result that can be specialized for use in any context. ...

12 Escape analysis: correctness proof, implementation and experimental results
Bruno Blanchet
January 1998 **Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Publisher: ACM Press
Full text available: pdf(1.68 MB) Additional Information: full citation, references, citings, index terms

13 Enabling frameworks and foundations part 1: technologies: Processing link structures and linkbases in the web's open world linking
François Bry, Michael Eckert
September 2005 **Proceedings of the sixteenth ACM conference on Hypertext and hypermedia HYPERTEXT '05**
Publisher: ACM Press
Full text available: pdf(201.52 KB) Additional Information: full citation, abstract, references, index terms

Hyperlinks are an essential feature of the World Wide Web, highly responsible for its success. XLink improves on HTML's linking capabilities in several ways. In particular, links after XLink can be "out-of-line" (i.e., not defined at a link source) and collected in (possibly several) linkbases, which considerably ease building complex link structures.Regarding its architecture as a distributed and open system, the Web differs significantly from traditional hypermedia systems. Modeling of link st ...

Keywords: XLink, hyperlink, link modeling and processing, linkbase

14 Incrementalized pointer and escape analysis
Frédéric Vivien, Martin Rinard

May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation PLDI '01**, Volume 36 Issue 5
Publisher: ACM Press

Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a new pointer and escape analysis. Instead of analyzing the whole program, the algorithm incrementally analyzes only those parts of the program that may deliver useful results. An analysis policy monitors the analysis results to direct the incremental investment of analysis resources to those parts of the program that offer the highest expected optimization return.

Our experimental results show that almost all of the objects are allocated at a small number of allocation sit ...

15 Open worlds: diary of roundUP presenter

Paul Diefenbach
May 2000 **ACM SIGGRAPH Computer Graphics**, Volume 34 Issue 2
Publisher: ACM Press
Full text available: pdf(295.58 KB) Additional Information: full citation, index terms

16 Access control analysis: Data-centric security: role analysis and role typestates

Vugranam C. Sreedhar
June 2006 **Proceedings of the eleventh ACM symposium on Access control models and technologies SACMAT '06**
Publisher: ACM Press
Full text available: pdf(270.98 KB) Additional Information: full citation, abstract, references, index terms

In J2EE and .NET roles are assigned to methods using external configuration files, called the deployment descriptors. Assigning roles to methods, although conceptually simple, in practice it is quite complicated. For instance, in order for a deployer to assign a role *r* to a method *m*, the deployer must understand the set of roles *R* that are assigned to each method *n* that can be invoked directly or indirectly from *m*, and that *r* has to be "consistently" ass ...

Keywords: RBAC, role analysis, role escape analysis, role typestates

17 OMEN: A strategy for testing object-oriented software

Amie L. Souter, Lori L. Pollock
August 2000 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2000 ACM SIGSOFT international symposium on Software testing and analysis ISSTA '00**, Volume 25 Issue 5
Publisher: ACM Press

Full text available: pdf(329.36 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a strategy for structural testing of object-oriented software systems with possibly unknown clients and unknown information about invoked methods. By exploiting the combined points-to and escape analysis developed for compiler optimization, our testing paradigm does not require a whole program representation to be in memory simultaneously for testing analysis. Potential effects from outside the component under test are easily identified and reported to the tester. As c ...

18

Parallel architectures: Automatic fence insertion for shared memory multiprocessing
Xing Fang, Jaejin Lee, Samuel P. Midkiff
June 2003 **Proceedings of the 17th annual international conference on Supercomputing**
Publisher: ACM Press
Full text available: pdf(327.89 KB) Additional Information: full citation, abstract, references, index terms

In general, the hardware memory consistency model in a multiprocessor system is not identical to the memory model at the programming language level. Consequently, the programming language memory model must be mapped onto the hardware memory model. Memory fence instructions can be inserted by the compiler where needed to accomplish this mapping. We have developed and implemented several fence insertion and optimization algorithms in our Pensieve compiler project. We present the different fence in ...

Keywords: Java, compilers, consistency models

19 Field analysis: getting useful and low-cost interprocedural information
Sanjay Ghemawat, Keith H. Randall, Daniel J. Scales
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5
Publisher: ACM Press
Full text available: pdf(686.96 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a new limited form of interprocedural analysis called field analysis that can be used by a compiler to reduce the costs of modern language features such as object-oriented programming, automatic memory management, and run-time checks required for type safety. Unlike many previous interprocedural analyses, our analysis is cheap, and does not require access to the entire program. Field analysis exploits the declared access restrictions placed on fields in a modul ...

20 Partial method compilation using dynamic profile information
John Whaley
October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '01**, Volume 36 Issue 11
Publisher: ACM Press
Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

The traditional tradeoff when performing dynamic compilation is that of fast compilation time versus fast code performance. Most dynamic compilation systems for Java perform selective compilation and/or optimization at a method granularity. This is the not the optimal granularity level. However, compiling at a sub-method granularity is thought to be too complicated to be practical. This paper describes a straightforward technique for performing compilation and optimizations at a finer, sub-metho ...

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2006 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

PORTAL
USPTO
Subscribe (Full Service) Register (Limited Service, Free) Login
Search: The ACM Digital Library The Guide
"escape analysis" "stack recovery"
SEARCH
THE ACM DIGITAL LIBRARY
Feedback Report a problem Satisfaction survey

Terms used **escape analysis stack recovery** Found **159** of **178,880**

Sort results by relevance
Display results expanded form
Save results to a Binder
Search Tips
Open results in a new window
Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 159 Result page: **1** 2 3 4 5 6 7 8 next

Relevance scale

1 Global Context Recovery: A New Strategy for Syntactic Error Recovery by Table-Drive Parsers

Ajit B. Pai, Richard B. Kieburtz
January 1980 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 2 Issue 1
Publisher: ACM Press

Full text available: pdf(1.59 MB) Additional Information: full citation, abstract, references, citings, index terms

Described is a method for syntactic error recovery that is compatible with deterministic parsing methods and that is able to recover from many errors more quickly than do other schemes because it performs global context recovery. The method relies on fiducial symbols, which are typically reserved key words of a language, to provide mileposts for error recovery. The method has been applied to LL(1) parsers, for which a detailed algorithm is given, and informally proved correct. The algorithm ...

2 Escape analysis for Java™: Theory and practice

Bruno Blanchet
November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6
Publisher: ACM Press

Full text available: pdf(684.21 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Escape analysis is a static analysis that determines whether the lifetime of data may exceed its static scope.This paper first presents the design and correctness proof of an escape analysis for Java™. This analysis is interprocedural, context sensitive, and as flow-sensitive as the static single assignment form. So, assignments to object fields are analyzed in a flow-insensitive manner. Since Java is an imperative language, the effect of assignments must be precisely determined. Thi ...

Keywords: Java, optimization, stack allocation, static analysis, synchronization elimination

3 Stack allocation and synchronization optimizations for Java using escape analysis

Jong-Deok Choi, Manish Gupta, Mauricio J. Serrano, Vugranam C. Sreedhar, Samuel P. Midkiff
November 2003 **ACM Transactions on Programming Languages and Systems**

(TOPLAS), Volume 25 Issue 6
Publisher: ACM Press

Full text available: pdf(632.85 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This article presents an *escape analysis* framework for Java to determine (1) if an object is not reachable after its method of creation returns, allowing the object to be allocated on the stack, and (2) if an object is reachable only from a single thread during its lifetime, allowing unnecessary synchronization operations on that object to be removed. We introduce a new program abstraction for escape analysis, the *connection graph*, that is used to establish reachability relationshi ...

Keywords: Connection graphs, escape analysis, points-to graph

4 Escape analysis on lists

Young Gil Park, Benjamin Goldberg
July 1992 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1992 conference on Programming language design and implementation PLDI '92**, Volume 27 Issue 7
Publisher: ACM Press

Full text available: pdf(1.08 MB) Additional Information: full citation, abstract, references, citings, index terms

Higher order functional programs constantly allocate objects dynamically. These objects are typically cons cells, closures, and records and are generally allocated in the heap and reclaimed later by some garbage collection process. This paper describes a compile time analysis, called escape analysis, for determining the lifetime of dynamically created objects in higher order functional programs, and describes optimizations that can be performed, based on the analysis, to improve storage all ...

5 Escape analysis for Java

Jong-Deok Choi, Manish Gupta, Mauricio Serrano, Vugranam C. Sreedhar, Sam Midkiff
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10
Publisher: ACM Press

Full text available: pdf(1.85 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a simple and efficient data flow algorithm for escape analysis of objects in Java programs to determine (i) if an object can be allocated on the stack; (ii) if an object is accessed only by a single thread during its lifetime, so that synchronization operations on that object can be removed. We introduce a new program abstraction for escape analysis, the connection graph, that is used to establish reachability relationships between objects and object ref ...

6 Escape analysis for object-oriented languages: application to Java

Bruno Blanchet
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10
Publisher: ACM Press

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

Escape analysis [27, 14, 5] is a static analysis that determines whether the lifetime of data exceeds its static scope.The main originality of our escape analysis is that it determines

precisely the effect of assignments, which is necessary to apply it to object oriented languages with promising results, whereas previous work [27, 14, 5] applied it to functional languages and were very imprecise on assignments. Our implementation analyses the full Java™ Language. ...

7 Dynamic compilation techniques: Escape analysis in the context of dynamic compilation and deoptimization

Thomas Kotzmann, Hanspeter Mössenböck

June 2005 **Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments**

Publisher: ACM Press

Full text available: pdf(203.40 KB) Additional Information: full citation, abstract, references, index terms

In object-oriented programming languages, an object is said to *escape* the method or thread in which it was created if it can also be accessed by other methods or threads. Knowing which objects do not escape allows a compiler to perform aggressive optimizations.This paper presents a new intraprocedural and interprocedural algorithm for escape analysis in the context of dynamic compilation where the compiler has to cope with dynamic class loading and deoptimization. It was implemented for S ...

Keywords: Java, deoptimization, escape analysis, just-in-time compilation, optimization, scalar replacement, stack allocation, synchronization removal

8 Compositional pointer and escape analysis for Java programs

John Whaley, Martin Rinard

October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10

Publisher: ACM Press

Full text available: pdf(2.42 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a combined pointer and escape analysis algorithm for Java programs. The algorithm is based on the abstraction of points-to escape graphs, which characterize how local variables and fields in objects refer to other objects. Each points-to escape graph also contains escape information, which characterizes how objects allocated in one region of the program can escape to be accessed by another region. The algorithm is designed to analyze arbitrary regions of complete or inco ...

9 Reference escape analysis: optimizing reference counting based on the lifetime of references

Young Gil Park, Benjamin Goldberg

May 1991 **ACM SIGPLAN Notices , Proceedings of the 1991 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation PEPM '91**, Volume 26 Issue 9

Publisher: ACM Press

Full text available: pdf(1.06 MB) Additional Information: full citation, references, citings, index terms

10 Pointer and escape analysis for multithreaded programs

Alexandru Salcianu, Martin Rinard

June 2001 **ACM SIGPLAN Notices , Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming PPoPP '01**, Volume 36 Issue 7

Publisher: ACM Press

Full text available: pdf(318.11 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a new combined pointer and escape analysis for multithreaded programs. The algorithm uses a new abstraction called parallel interaction graphs to analyze the interactions between threads and extract precise points-to, escape, and action ordering information for objects accessed by multiple threads. The analysis is compositional, analyzing each method or thread once to extract a parameterized analysis result that can be specialized for use in any context. ...

11 Escape analysis: correctness proof, implementation and experimental results

Bruno Blanchet

January 1998 **Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Publisher: ACM Press

Full text available: pdf(1.68 MB) Additional Information: full citation, references, citings, index terms

12 Incrementalized pointer and escape analysis

Frédéric Vivien, Martin Rinard

May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation PLDI '01**, Volume 36 Issue 5

Publisher: ACM Press

Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a new pointer and escape analysis. Instead of analyzing the whole program, the algorithm incrementally analyzes only those parts of the program that may deliver useful results. An analysis policy monitors the analysis results to direct the incremental investment of analysis resources to those parts of the program that offer the highest expected optimization return.

Our experimental results show that almost all of the objects are allocated at a small number of allocation sit ...

13 Access control analysis: Data-centric security: role analysis and role typestates

Vugranam C. Sreedhar

June 2006 **Proceedings of the eleventh ACM symposium on Access control models and technologies SACMAT '06**

Publisher: ACM Press

Full text available: pdf(270.98 KB) Additional Information: full citation, abstract, references, index terms

In J2EE and .NET roles are assigned to methods using external configuration files, called the deployment descriptors. Assigning roles to methods, although conceptually simple, in practice it is quite complicated. For instance, in order for a deployer to assign a role *r* to a method *m*, the deployer must understand the set of roles *R* that are assigned to each method *n* that can be invoked directly or indirectly from *m*, and that *r* has to be "consistently" ass ...

Keywords: RBAC, role analysis, role escape analysis, role typestates

14 OMEN: A strategy for testing object-oriented software

Amie L. Souter, Lori L. Pollock

August 2000 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2000**

ACM SIGSOFT international symposium on Software testing and analysis ISSTA '00, Volume 25 Issue 5
Publisher: ACM Press

Full text available: pdf(329.36 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a strategy for structural testing of object-oriented software systems with possibly unknown clients and unknown information about invoked methods. By exploiting the combined points-to and escape analysis developed for compiler optimization, our testing paradigm does not require a whole program representation to be in memory simultaneously for testing analysis. Potential effects from outside the component under test are easily identified and reported to the tester. As c ...

15 Parallel architectures: Automatic fence insertion for shared memory multiprocessing

Xing Fang, Jaejin Lee, Samuel P. Midkiff
June 2003 **Proceedings of the 17th annual international conference on Supercomputing**
Publisher: ACM Press

Full text available: pdf(327.89 KB) Additional Information: full citation, abstract, references, index terms

In general, the hardware memory consistency model in a multiprocessor system is not identical to the memory model at the programming language level. Consequently, the programming language memory model must be mapped onto the hardware memory model. Memory fence instructions can be inserted by the compiler where needed to accomplish this mapping. We have developed and implemented several fence insertion and optimization algorithms in our Pensieve compiler project. We present the different fence in ...

Keywords: Java, compilers, consistency models

16 Field analysis: getting useful and low-cost interprocedural information

Sanjay Ghemawat, Keith H. Randall, Daniel J. Scales
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5
Publisher: ACM Press

Full text available: pdf(686.96 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a new limited form of interprocedural analysis called field analysis that can be used by a compiler to reduce the costs of modern language features such as object-oriented programming, automatic memory management, and run-time checks required for type safety. Unlike many previous interprocedural analyses, our analysis is cheap, and does not require access to the entire program. Field analysis exploits the declared access restrictions placed on fields in a modul ...

17 Partial method compilation using dynamic profile information

John Whaley
October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '01**, Volume 36 Issue 11
Publisher: ACM Press

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

The traditional tradeoff when performing dynamic compilation is that of fast compilation time versus fast code performance. Most dynamic compilation systems for Java perform

selective compilation and/or optimization at a method granularity. This is the not the optimal granularity level. However, compiling at a sub-method granularity is thought to be too complicated to be practical. This paper describes a straightforward technique for performing compilation and optimizations at a finer, sub-metho ...

18 Compiler optimizations for power, performance: Tracking object life cycle for leakage energy optimization

G. Chen, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, M. Wolczko

October 2003 **Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis**

Publisher: ACM Press

Full text available: pdf(304.66 KB) Additional Information: full citation, abstract, references, index terms

The focus of this work is on utilizing the state of objects during their lifespan in optimizing the leakage energy consumed in the data caches when executing embedded Java applications. Our analysis reveals that a major portion of the leakage energy is actually wasted in retaining the objects beyond their last use. In order to eliminate this wastage, we investigate three approaches that use the garbage collector, escape analysis and last use analysis for reducing leakage energy. Finally, we trac ...

Keywords: Java, cache, leakage energy

19 A region-based compilation technique for a Java just-in-time compiler

Toshio Suganuma, Toshiaki Yasue, Toshio Nakatani

May 2003 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation PLDI '03**, Volume 38 Issue 5

Publisher: ACM Press

Full text available: pdf(158.62 KB) Additional Information: full citation, abstract, references, citings, index terms

Method inlining and data flow analysis are two major optimization components for effective program transformations, however they often suffer from the existence of rarely or never executed code contained in the target method. One major problem lies in the assumption that the compilation unit is partitioned at method boundaries. This paper describes the design and implementation of a region-based compilation technique in our dynamic compilation system, in which the compiled regions are selected a ...

Keywords: dynamic compilers, on-stack replacement, partial inlining, region-based compilation

20 Cloning-based context-sensitive pointer alias analysis using binary decision diagrams

John Whaley, Monica S. Lam

June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation PLDI '04**, Volume 39 Issue 6

Publisher: ACM Press

Full text available: pdf(277.87 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents the first scalable context-sensitive, inclusion-based pointer alias analysis for Java programs. Our approach to context sensitivity is to create a clone of a method for every context of interest, and run a *context-insensitive* algorithm over the expanded call graph to get *context-sensitive* results. For precision, we generate a clone for every acyclic path through a program's call graph, treating methods in a strongly connected component as a single node. Normally ...

Keywords: Datalog, Java, binary decision diagrams, cloning, context-sensitive, inclusion-based, logic programming, pointer analysis, program analysis, scalable

Results 1 - 20 of 159 Result page: **1** 2 3 4 5 6 7 8 next

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2006 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

PORTAL
USPTO
Subscribe (Full Service) Register (Limited Service, Free) Login
Search: The ACM Digital Library The Guide
"escape analysis" and "compensation publishing"
SEARCH
THE ACM DIGITAL LIBRARY
Feedback Report a problem Satisfaction survey

Terms used **escape analysis** and **compensation publishing** Found **158** of **178,880**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 158 Result page: **1** 2 3 4 5 6 7 8 next

Relevance scale

1 Escape analysis for Java™: Theory and practice

Bruno Blanchet
November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6
Publisher: ACM Press

Full text available: pdf(684.21 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Escape analysis is a static analysis that determines whether the lifetime of data may exceed its static scope.This paper first presents the design and correctness proof of an escape analysis for Java™. This analysis is interprocedural, context sensitive, and as flow-sensitive as the static single assignment form. So, assignments to object fields are analyzed in a flow-insensitive manner. Since Java is an imperative language, the effect of assignments must be precisely determined. Thi ...

Keywords: Java, optimization, stack allocation, static analysis, synchronization elimination

2 Stack allocation and synchronization optimizations for Java using escape analysis

Jong-Deok Choi, Manish Gupta, Mauricio J. Serrano, Vugranam C. Sreedhar, Samuel P. Midkiff
November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6
Publisher: ACM Press

Full text available: pdf(632.85 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This article presents an *escape analysis* framework for Java to determine (1) if an object is not reachable after its method of creation returns, allowing the object to be allocated on the stack, and (2) if an object is reachable only from a single thread during its lifetime, allowing unnecessary synchronization operations on that object to be removed. We introduce a new program abstraction for escape analysis, the *connection graph*, that is used to establish reachability relationshi ...

Keywords: Connection graphs, escape analysis, points-to graph

3 Escape analysis on lists

Young Gil Park, Benjamin Goldberg
July 1992 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1992 conference on Programming language design and implementation PLDI '92**, Volume 27 Issue 7
Publisher: ACM Press

Full text available: pdf(1.08 MB) Additional Information: full citation, abstract, references, citings, index terms

Higher order functional programs constantly allocate objects dynamically. These objects are typically cons cells, closures, and records and are generally allocated in the heap and reclaimed later by some garbage collection process. This paper describes a compile time analysis, called escape analysis, for determining the lifetime of dynamically created objects in higher order functional programs, and describes optimizations that can be performed, based on the analysis, to improve storage all ...

4 Escape analysis for Java

Jong-Deok Choi, Manish Gupta, Mauricio Serrano, Vugranam C. Sreedhar, Sam Midkiff
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10
Publisher: ACM Press

Full text available: pdf(1.85 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a simple and efficient data flow algorithm for escape analysis of objects in Java programs to determine (i) if an object can be allocated on the stack; (ii) if an object is accessed only by a single thread during its lifetime, so that synchronization operations on that object can be removed. We introduce a new program abstraction for escape analysis, the connection graph, that is used to establish reachability relationships between objects and object ref ...

5 Escape analysis for object-oriented languages: application to Java

Bruno Blanchet
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10
Publisher: ACM Press

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

Escape analysis [27, 14, 5] is a static analysis that determines whether the lifetime of data exceeds its static scope.The main originality of our escape analysis is that it determines precisely the effect of assignments, which is necessary to apply it to object oriented languages with promising results, whereas previous work [27, 14, 5] applied it to functional languages and were very imprecise on assignments. Our implementation analyses the full Java™ Language. ...

6 Dynamic compilation techniques: Escape analysis in the context of dynamic compilation and deoptimization

Thomas Kotzmann, Hanspeter Mössenböck
June 2005 **Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments**
Publisher: ACM Press

Full text available: pdf(203.40 KB) Additional Information: full citation, abstract, references, index terms

In object-oriented programming languages, an object is said to *escape* the method or thread in which it was created if it can also be accessed by other methods or threads. Knowing which objects do not escape allows a compiler to perform aggressive

optimizations.This paper presents a new intraprocedural and interprocedural algorithm for escape analysis in the context of dynamic compilation where the compiler has to cope with dynamic class loading and deoptimization. It was implemented for S ...

Keywords: Java, deoptimization, escape analysis, just-in-time compilation, optimization, scalar replacement, stack allocation, synchronization removal

7 Compositional pointer and escape analysis for Java programs

John Whaley, Martin Rinard

October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '99**, Volume 34 Issue 10

Publisher: ACM Press

Full text available: pdf(2.42 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a combined pointer and escape analysis algorithm for Java programs. The algorithm is based on the abstraction of points-to escape graphs, which characterize how local variables and fields in objects refer to other objects. Each points-to escape graph also contains escape information, which characterizes how objects allocated in one region of the program can escape to be accessed by another region. The algorithm is designed to analyze arbitrary regions of complete or inco ...

8 Reference escape analysis: optimizing reference counting based on the lifetime of references

Young Gil Park, Benjamin Goldberg

May 1991 **ACM SIGPLAN Notices , Proceedings of the 1991 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation PEPM '91**, Volume 26 Issue 9

Publisher: ACM Press

Full text available: pdf(1.06 MB) Additional Information: full citation, references, citings, index terms

9 Pointer and escape analysis for multithreaded programs

Alexandru Salcianu, Martin Rinard

June 2001 **ACM SIGPLAN Notices , Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming PPoPP '01**, Volume 36 Issue 7

Publisher: ACM Press

Full text available: pdf(318.11 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a new combined pointer and escape analysis for multithreaded programs. The algorithm uses a new abstraction called parallel interaction graphs to analyze the interactions between threads and extract precise points-to, escape, and action ordering information for objects accessed by multiple threads. The analysis is compositional, analyzing each method or thread once to extract a parameterized analysis result that can be specialized for use in any context. ...

10 Escape analysis: correctness proof, implementation and experimental results

Bruno Blanchet

January 1998 **Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Publisher: ACM Press

Full text available: pdf(1.68 MB) Additional Information: full citation, references, citings, index terms

11 Incrementalized pointer and escape analysis

Frédéric Vivien, Martin Rinard

May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation PLDI '01**, Volume 36 Issue 5

Publisher: ACM Press

Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a new pointer and escape analysis. Instead of analyzing the whole program, the algorithm incrementally analyzes only those parts of the program that may deliver useful results. An analysis policy monitors the analysis results to direct the incremental investment of analysis resources to those parts of the program that offer the highest expected optimization return.

Our experimental results show that almost all of the objects are allocated at a small number of allocation sit ...

12 Access control analysis: Data-centric security: role analysis and role typestates

Vugranam C. Sreedhar

June 2006 **Proceedings of the eleventh ACM symposium on Access control models and technologies SACMAT '06**

Publisher: ACM Press

Full text available: pdf(270.98 KB) Additional Information: full citation, abstract, references, index terms

In J2EE and .NET roles are assigned to methods using external configuration files, called the deployment descriptors. Assigning roles to methods, although conceptually simple, in practice it is quite complicated. For instance, in order for a deployer to assign a role *r* to a method *m*, the deployer must understand the set of roles *R* that are assigned to each method *n* that can be invoked directly or indirectly from *m*, and that *r* has to be "consistently" ass ...

Keywords: RBAC, role analysis, role escape analysis, role typestates

13 OMEN: A strategy for testing object-oriented software

Amie L. Souter, Lori L. Pollock

August 2000 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2000 ACM SIGSOFT international symposium on Software testing and analysis ISSTA '00**, Volume 25 Issue 5

Publisher: ACM Press

Full text available: pdf(329.36 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a strategy for structural testing of object-oriented software systems with possibly unknown clients and unknown information about invoked methods. By exploiting the combined points-to and escape analysis developed for compiler optimization, our testing paradigm does not require a whole program representation to be in memory simultaneously for testing analysis. Potential effects from outside the component under test are easily identified and reported to the tester. As c ...

14 Parallel architectures: Automatic fence insertion for shared memory multiprocessing

Xing Fang, Jaejin Lee, Samuel P. Midkiff

June 2003 **Proceedings of the 17th annual international conference on Supercomputing**

Publisher: ACM Press
Full text available: pdf(327.89 KB) Additional Information: full citation, abstract, references, index terms

In general, the hardware memory consistency model in a multiprocessor system is not identical to the memory model at the programming language level. Consequently, the programming language memory model must be mapped onto the hardware memory model. Memory fence instructions can be inserted by the compiler where needed to accomplish this mapping. We have developed and implemented several fence insertion and optimization algorithms in our Pensieve compiler project. We present the different fence in ...

Keywords: Java, compilers, consistency models

15 Field analysis: getting useful and low-cost interprocedural information
Sanjay Ghemawat, Keith H. Randall, Daniel J. Scales
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5
Publisher: ACM Press
Full text available: pdf(686.96 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a new limited form of interprocedural analysis called field analysis that can be used by a compiler to reduce the costs of modern language features such as object-oriented programming, automatic memory management, and run-time checks required for type safety. Unlike many previous interprocedural analyses, our analysis is cheap, and does not require access to the entire program. Field analysis exploits the declared access restrictions placed on fields in a modul ...

16 Partial method compilation using dynamic profile information
John Whaley
October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '01**, Volume 36 Issue 11
Publisher: ACM Press
Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

The traditional tradeoff when performing dynamic compilation is that of fast compilation time versus fast code performance. Most dynamic compilation systems for Java perform selective compilation and/or optimization at a method granularity. This is the not the optimal granularity level. However, compiling at a sub-method granularity is thought to be too complicated to be practical. This paper describes a straightforward technique for performing compilation and optimizations at a finer, sub-metho ...

17 Compiler optimizations for power, performance: Tracking object life cycle for leakage energy optimization
G. Chen, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, M. Wolczko
October 2003 **Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis**
Publisher: ACM Press
Full text available: pdf(304.66 KB) Additional Information: full citation, abstract, references, index terms

The focus of this work is on utilizing the state of objects during their lifespan in optimizing the leakage energy consumed in the data caches when executing embedded Java applications. Our analysis reveals that a major portion of the leakage energy is actually wasted in retaining the objects beyond their last use. In order to eliminate this wastage,

we investigate three approaches that use the garbage collector, escape analysis and last use analysis for reducing leakage energy. Finally, we trac ...

Keywords: Java, cache, leakage energy

18 A region-based compilation technique for a Java just-in-time compiler

Toshio Suganuma, Toshiaki Yasue, Toshio Nakatani

May 2003 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation PLDI '03**, Volume 38 Issue 5

Publisher: ACM Press

Full text available: pdf(158.62 KB) Additional Information: full citation, abstract, references, citings, index terms

Method inlining and data flow analysis are two major optimization components for effective program transformations, however they often suffer from the existence of rarely or never executed code contained in the target method. One major problem lies in the assumption that the compilation unit is partitioned at method boundaries. This paper describes the design and implementation of a region-based compilation technique in our dynamic compilation system, in which the compiled regions are selected a ...

Keywords: dynamic compilers, on-stack replacement, partial inlining, region-based compilation

19 Cloning-based context-sensitive pointer alias analysis using binary decision diagrams

John Whaley, Monica S. Lam

June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation PLDI '04**, Volume 39 Issue 6

Publisher: ACM Press

Full text available: pdf(277.87 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents the first scalable context-sensitive, inclusion-based pointer alias analysis for Java programs. Our approach to context sensitivity is to create a clone of a method for every context of interest, and run a *context-insensitive* algorithm over the expanded call graph to get *context-sensitive* results. For precision, we generate a clone for every acyclic path through a program's call graph, treating methods in a strongly connected component as a single node. Normally ...

Keywords: Datalog, Java, binary decision diagrams, cloning, context-sensitive, inclusion-based, logic programming, pointer analysis, program analysis, scalable

20 Effective synchronization removal for Java

Erik Ruf

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5

Publisher: ACM Press

Full text available: pdf(819.53 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a new technique for removing unnecessary synchronization operations from statically compiled Java programs. Our approach improves upon current efforts based on escape analysis, as it can eliminate synchronization operations even on objects that

escape their allocating threads. It makes use of a compact, equivalence-class-based representation that eliminates the need for fixed point operations during the analysis. We describe and evaluate the performance of an implemen ...

Results 1 - 20 of 158 Result page: **1** 2 3 4 5 6 7 8 next

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2006 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.